SDC

STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2015

Where Moore's law meets the speed of light: optimizing Exabyte-scale network protocols

Yogesh Vedpathak

Cleversafe Inc.

Topics

- Exabyte Storage System
- Design Goals for Scalable System
- Limitations of Today's Protocols
- Designing Protocol for Tomorrow's Storage Systems

Storage Trends

- It's a no surprise that the data is growing

Data Phase	Astronomy	Twitter	YouTube	Genomics
Acquisition	25 zetta-bytes/year	0.5–15 billion tweets/year	500–900 million hours/year	1 zetta-bases/year
Storage	1 EB/year	1–17 PB/year	1–2 EB/year	2–40 EB/year
Analysis	In situ data reduction	Topic and sentiment mining	Limited requirements	Heterogeneous data and analysis
	Real-time processing	Metadata analysis		Variant calling, ~2 trillion central processing unit (CPU) hours
	Massive volumes			All-pairs genome alignments, ~10,000 trillion CPU hours
Distribution	Dedicated lines from antennae to server (600 TB/s)	Small units of distribution	Major component of modern user's bandwidth (10 MB/s)	Many small (10 MB/s) and fewer massive (10 TB/s) data movement

doi:10.1371/journal.pbio.1002195.t001

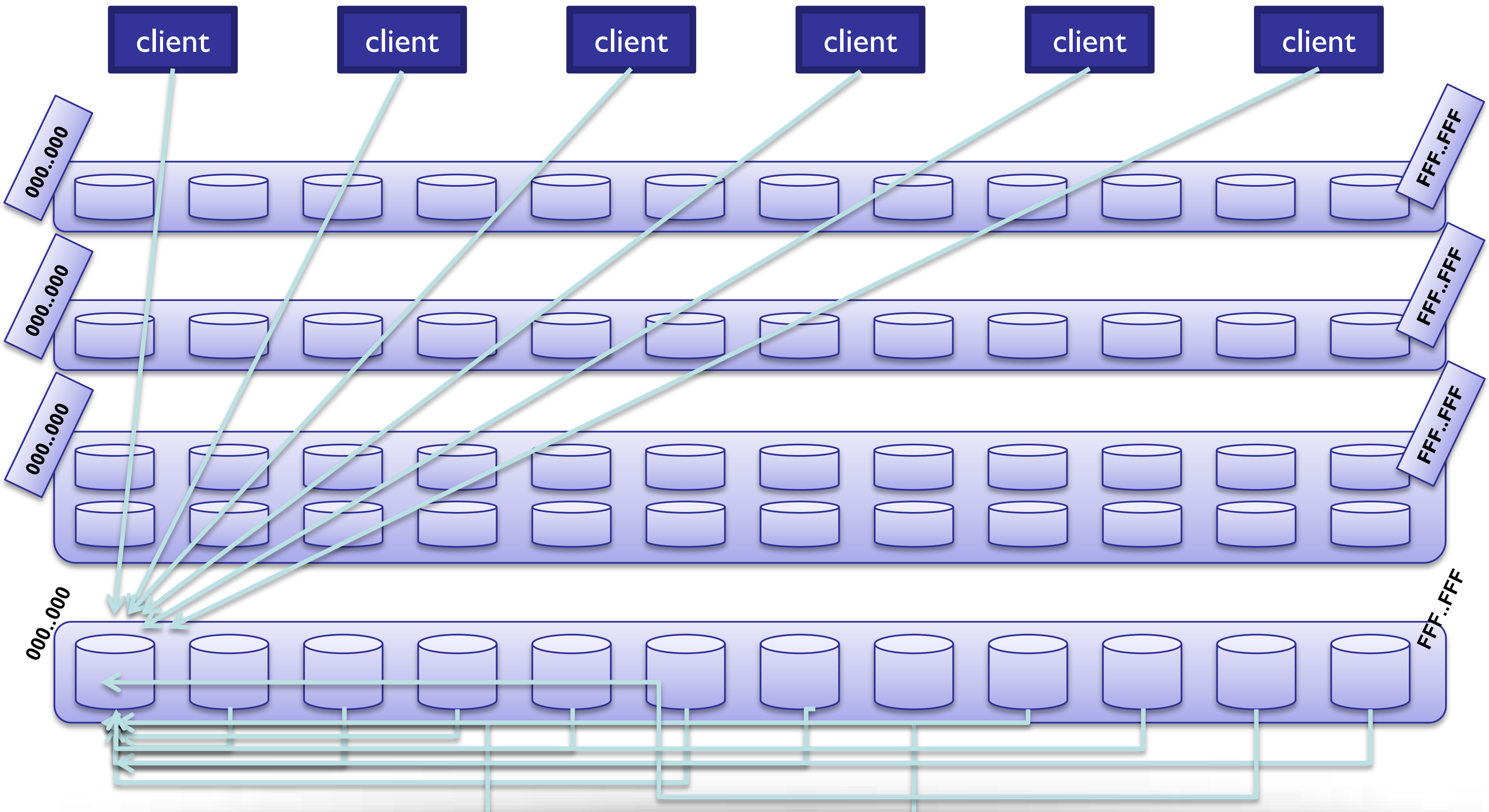
What does 10EB look like?

- 2,621,440 hard drives (4TB)
  - 10,923 drives will fail each month (5% AFR)
- 54,613 storage nodes (48 drives each)
- 3 geo-dispersed sites
- 5,462 racks or 1,820 at each site

Design Goals of Scalable System

- Support Internet-scale systems having no inherent or theoretical limits
- Suffer zero degradation in latency, OPS or throughput as system grows
- Enable decentralized access by an unlimited number of readers and writers

Components of Scalable Storage System



Storage Node Scalability Challenge

- A client may need to communicate with thousands of storage nodes at once
- A storage node needs to accept incoming connection from
  - Client performing IO
  - Another storage node for rebuilding

Today’s Protocols Have Hit the Limit

- TCP/TLS are slow to start
  - 3-way handshake: 1 RT before sending data
  - TLS negotiation: 3 RT before sending data
- Congestion control hurts more than help
  - 1 Packet loss slows down entire stream
- No prioritization of data once sent on the wire

TCP 3-way Handshake

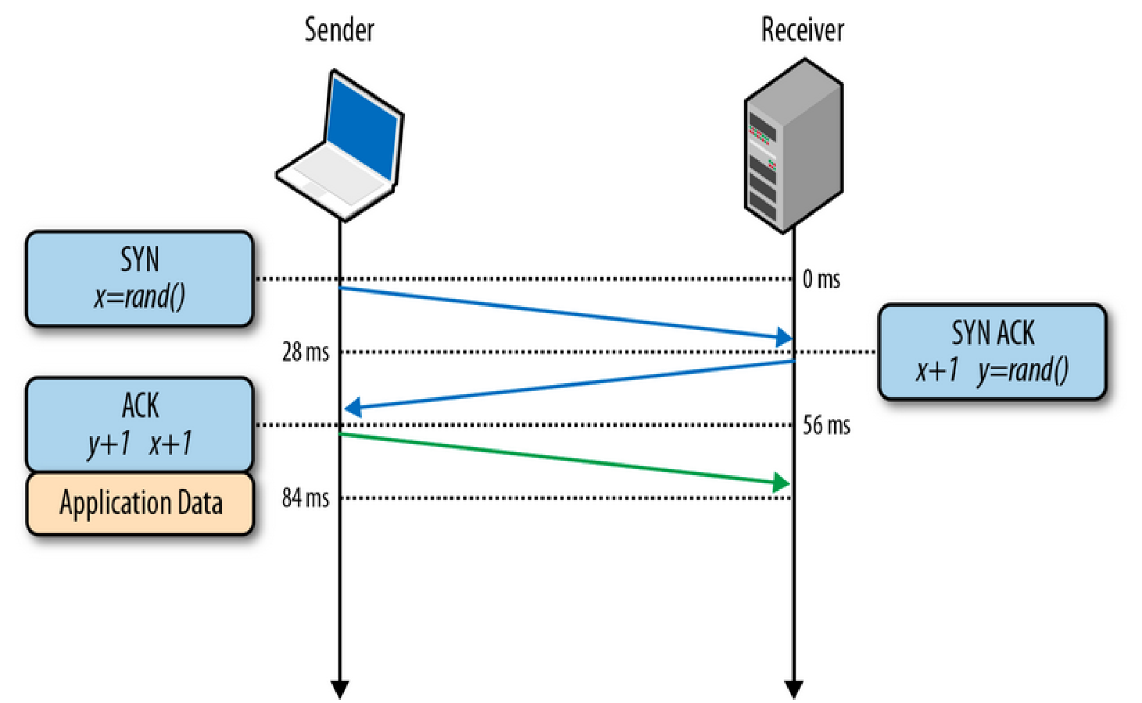


Figure 2-1. Three-way handshake

TLS Negotiation Protocol

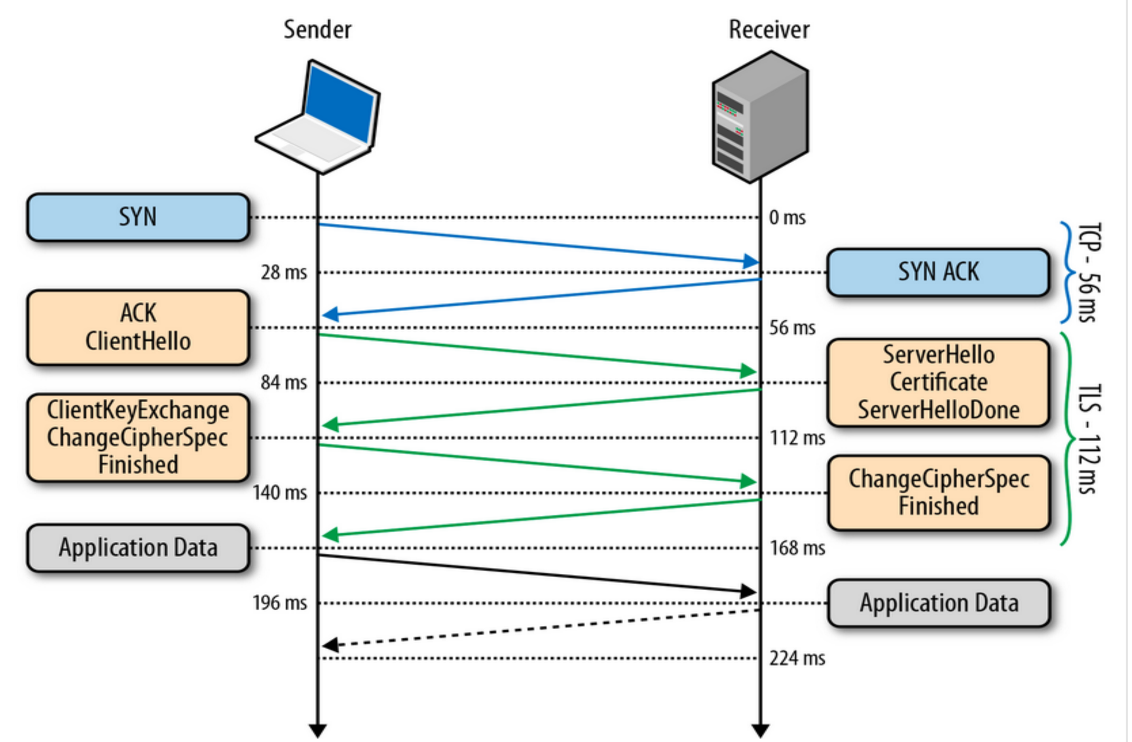


Figure 4-2. TLS handshake protocol

Internet Protocols Evolve Slowly

- Ongoing efforts to latency problems
  - TLS 1.3
  - QUIC protocol
- The problem is they evolve slowly and it takes time to deploy them
  - On both ends
  - On middle boxes

It’s All About the Latency

- Throughput is important but latency matters more for object based transfer
- Connection setup latency
  - 0-RTT/1-RTT connection setup
- Response latency
  - Multiplexing
  - Event driver implementation
  - Data prioritization

0-RTT Connection Setup

- Self validating message/request

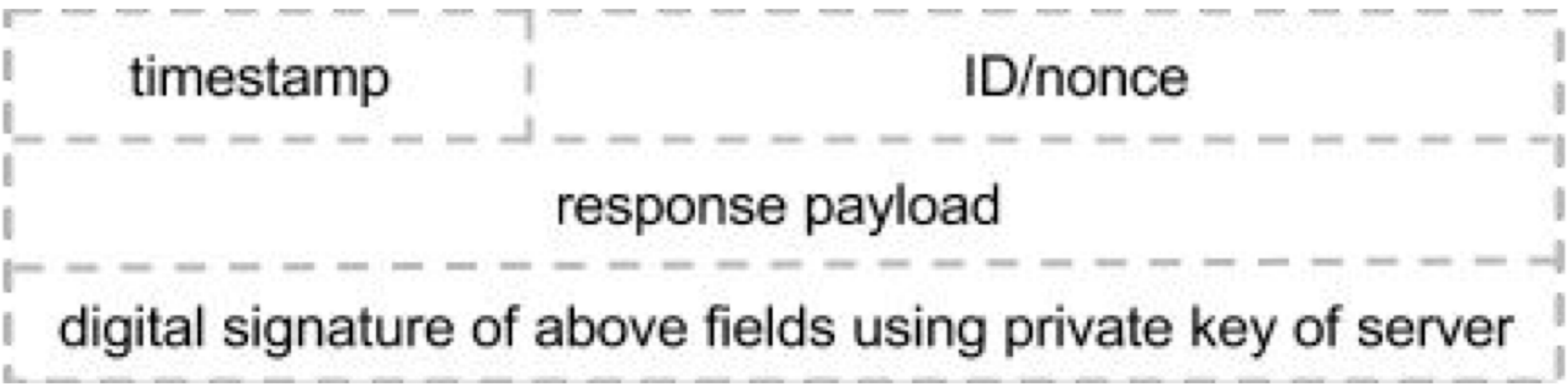
timestamp	ID/nonce
random master key encrypted with server's public key	
certificate chain of the client	
request payload	
digital signature of above fields using private key of the client	

Processing Self Validating Requests

- Verify request is in pre-defined time window
- Verify that the request is not repeated
- Verify that the client's certificate chain is valid and no certificate is revoked or expired
- Verify that the signature is valid
- Decrypt random master key using private key of the server
- Use client's master key to decrypt the message

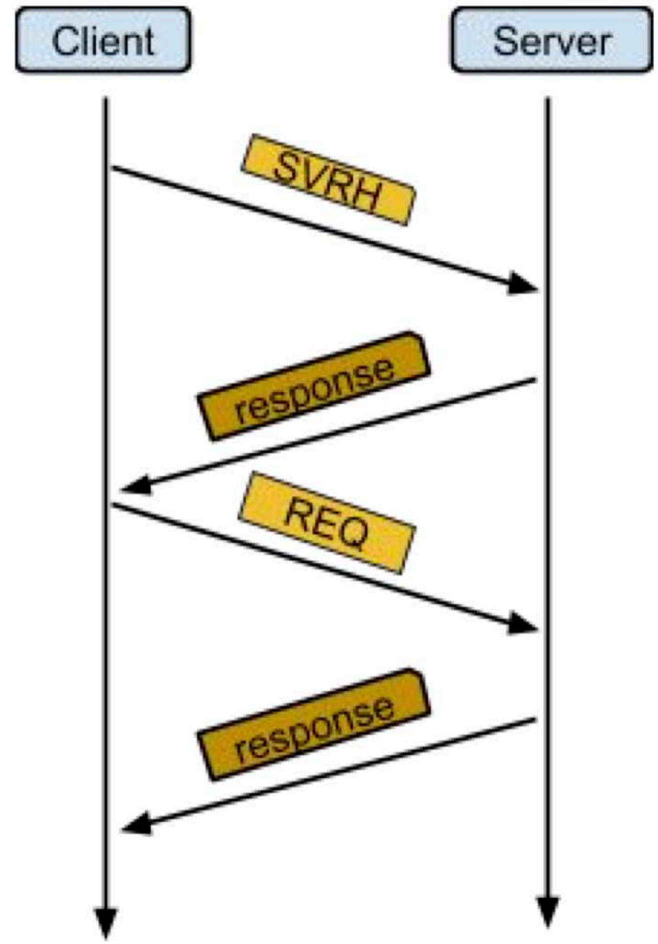
0-RTT Connection Setup

- Self validating response



Processing Self Validating Response

- The response is then encrypted with server's key
- Sign with server's signing key
- The client validate the signature of the response

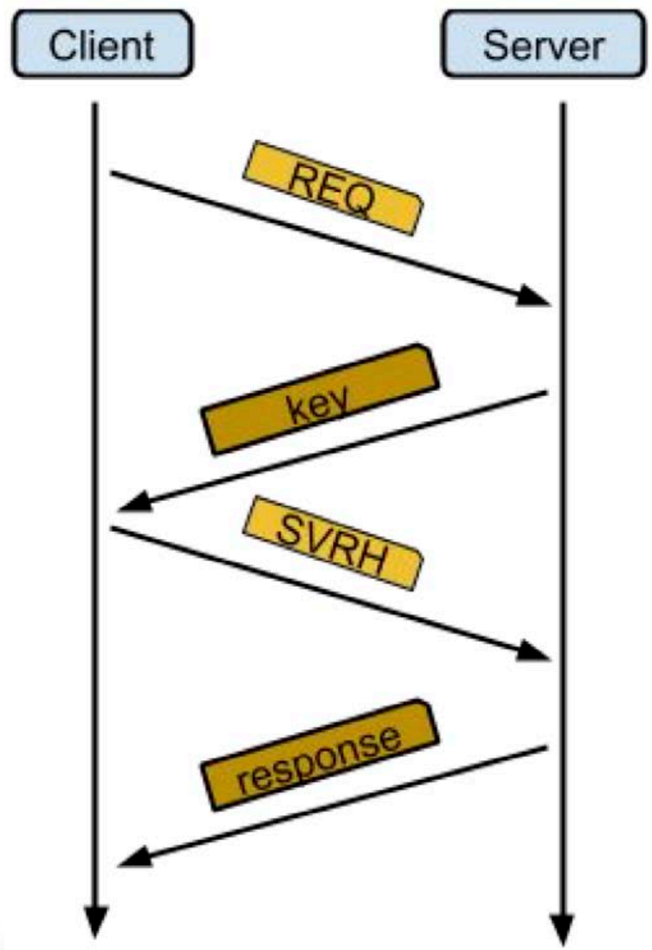


1-RTT Connection Setup

- The previous example assumes that the client has public key/certificate of the server
- Additional RT is required if:
  - Client does not have server's public key
  - Client has a wrong/expired public key of server

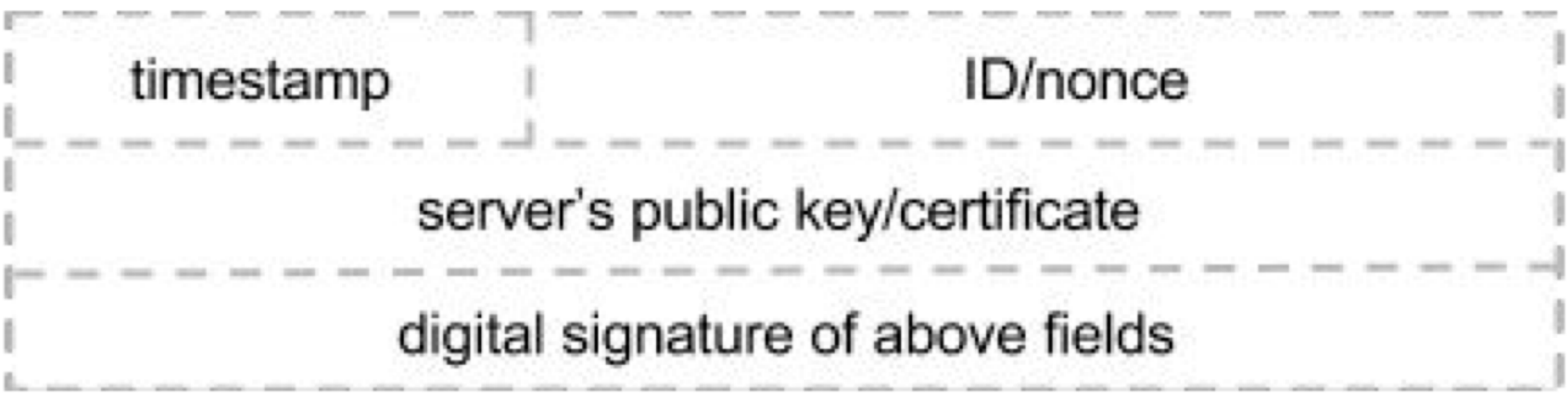
1-RTT Connection Setup

- Client send simple request with nonce/ID
- Send back the public key/certificate change in response
- Client then send SVRH with payload



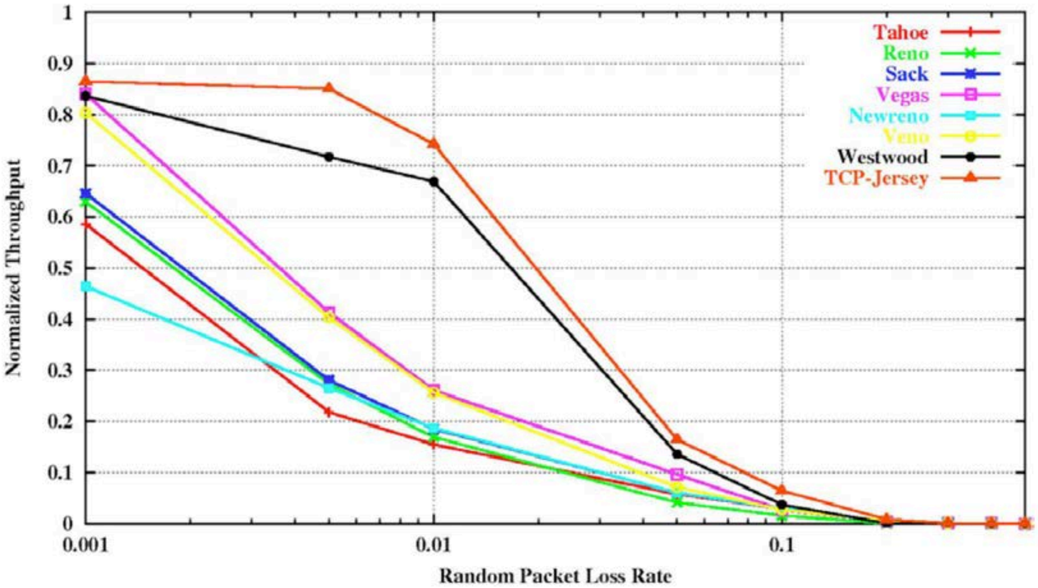
1-RTT Connection

- Server response with public key



Improving Response Latency

- Congestion Control in TCP may cause packet loss
- Slow start limits in-flight data to congestion window
- How to address these issues?

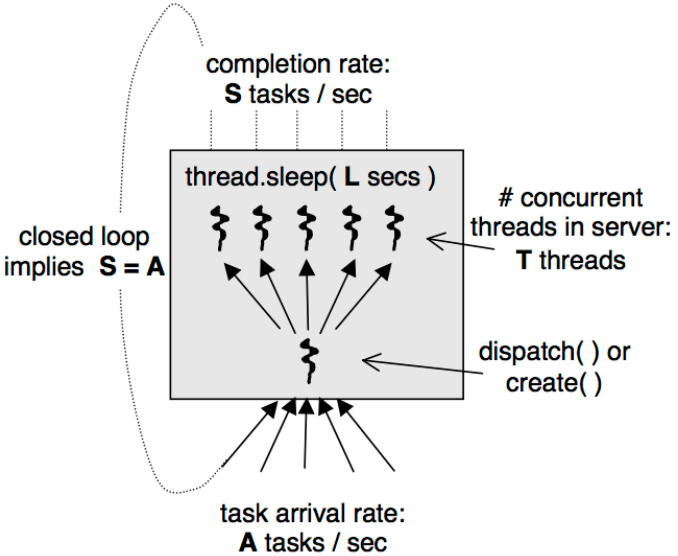


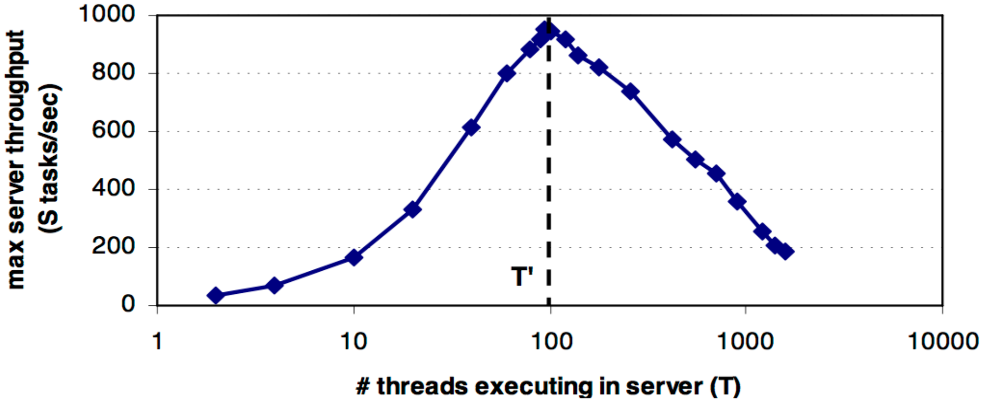
Multiplexing

- TCP is a stream protocol but UDP is not
- Application can use UPD and convert discrete messages into a stream
- Multiplexing is maintaining a session between client and server
- Sessions are allows
  - Correct ordering of the messages
  - Having more than one process on same “box” communicate with server

Event Driven Implementation

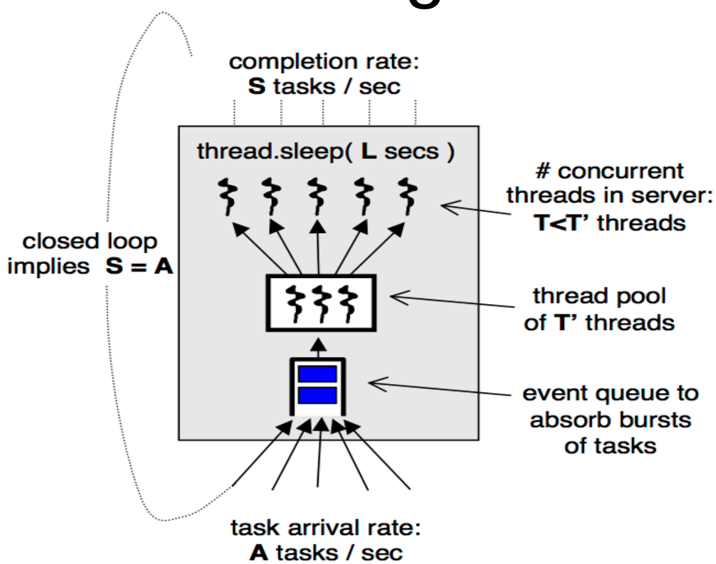
- One thread per connection
  - Context switching is expensive

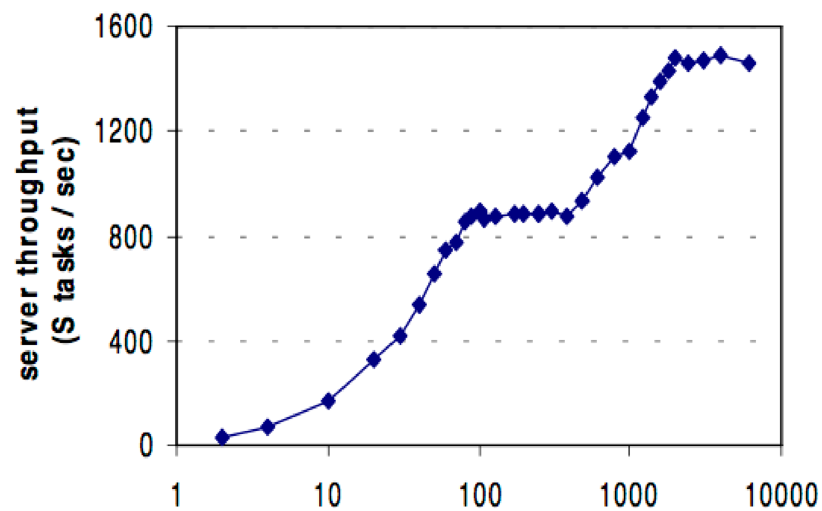




Event Driven Implementation

- Application uses constant size thread pool
- A thread is selected to execute tasks
- Remaining tasks are queued





Data Prioritization

- Once data is sent on the wire it can't be prioritized
- Imaging the client is writing a huge object. Meaning client is writing data on wire as fast as it can
- What if there is read or lookup request for another object comes in?

Data Prioritization

- Massage based transfer over UPD come to rescue again
- Each message has a priority
- Client application sends only small chunk of data at a time
- Rest of the messages are kept in memory in their priority order

Conclusion

- Need for data transfer over internet is increasing
- Today's transfer protocols suffer high latency
- Low latency, secure protocol are possible without requiring infrastructural changes

Questions